**1. Describing the Filter API**

A **Filter** is a component in Java web applications used to **intercept HTTP requests and responses**. Filters allow you to:

* Perform **pre-processing** (e.g., logging, authentication, input validation).
* Perform **post-processing** (e.g., compressing response, modifying headers).
* Work **independently of servlets or JSPs**.

Filters are defined in the javax.servlet (or jakarta.servlet in newer versions) package.

**✅ Common Use Cases**

* Logging user activity
* Authentication & authorization
* Input validation & sanitization
* Compression (e.g., GZIP)
* Setting character encoding

**🔨 2. Developing a Filter Class**

You must implement the javax.servlet.Filter interface and override three methods:

**🔸 Filter Interface Methods:**

public interface Filter {

void init(FilterConfig filterConfig) throws ServletException;

void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

throws IOException, ServletException;

void destroy();

}

**✅ Example: LoggingFilter.java**

import javax.servlet.\*;

import javax.servlet.annotation.WebFilter;

import java.io.IOException;

import java.util.Date;

@WebFilter("/hello") // Optional if configured in web.xml

public class LoggingFilter implements Filter {

public void init(FilterConfig config) throws ServletException {

System.out.println("LoggingFilter initialized");

}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

throws IOException, ServletException {

System.out.println("Request received at: " + new Date());

// Pass the request along the filter chain

chain.doFilter(request, response);

System.out.println("Response sent at: " + new Date());

}

public void destroy() {

System.out.println("LoggingFilter destroyed");

}

}

**⚙️ 3. Configuring a Filter in web.xml**

Even though annotations (@WebFilter) are common, configuring in web.xml offers finer control in large deployments.

**✅ Example web.xml Filter Configuration**

xml

<filter>

<filter-name>LoggingFilter</filter-name>

<filter-class>LoggingFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>LoggingFilter</filter-name>

<url-pattern>/hello</url-pattern>

</filter-mapping>

**🔁 Request Flow with a Filter**

scss

Client → [Filter] → Servlet → Response → [Filter] (post-processing) → Client

**✅ Output Example (on console)**

LoggingFilter initialized

Request received at: Sun Jun 2 15:34:00 IST 2025

Response sent at: Sun Jun 2 15:34:00 IST 2025

**Example: Configure a Filter for Multiple URLs in web.xml**

**🔧 web.xml Configuration**

<filter>

<filter-name>LoggingFilter</filter-name>

<filter-class>com.example.LoggingFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>LoggingFilter</filter-name>

<url-pattern>/hello</url-pattern>

<url-pattern>/admin/\*</url-pattern>

<url-pattern>/user/\*</url-pattern>

</filter-mapping>